

ЗМІСТ

ВСТУП	3
РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ	5
1.1 Керування семисегментним індикатором.	5
1.2 Робота з терміналом середовища “Intel FPGA Monitor Program”	6
РОЗДІЛ 2. СТРУКТУРНА СХЕМА ПРИСТРОЮ.....	7
2.1 Структурна схема пристрою та її опис	7
РОЗДІЛ 3. СТВОРЕННЯ ПРИСТРОЮ	9
3.1. Створення та симуляція модуля «Indicator Driver»	9
3.2. Проєкт в середовищі «Quartus»	11
3.3. Проєкт в середовищі «Intel FPGA Monitor Program»	15
ВИСНОВОК	17
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	18

					ДКЗ1мп. 466539.001 ПЗ	Лист
						2
Зм.	Лист	№ докум.	Підпис	Дата		

ВСТУП

В межах даної курсової роботи було вирішено розробити систему, що би дозволила користувачу керувати сукупністю семисегментних індикаторів через термінал комп'ютера.

Програмний термінал – це широко розповсюджений інструмент роботи з комп'ютерними чи програмними системами. Ось для приколу вигляд терміналу, що надається операційною системою “Windows” (Рис. 1) чи програмного забезпечення “Intel FPGA Monitor Program” (Рис. 2).

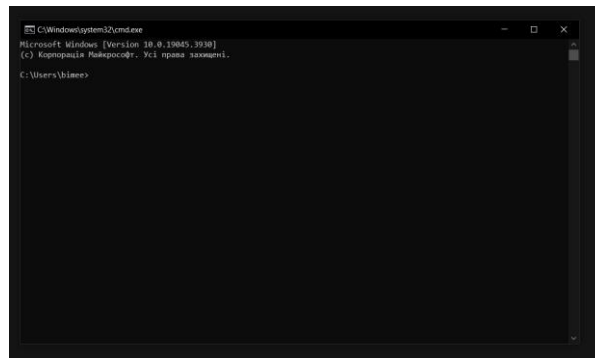


Рис. 1. Програмний термінал в операційній системі “Windows».

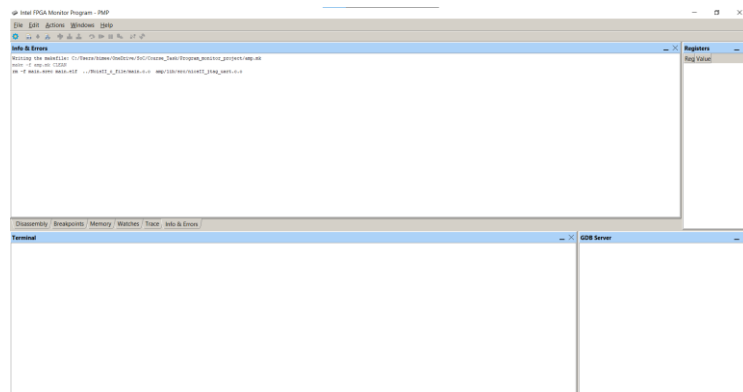


Рис. 2. Програмний термінал в операційній системі «Intel FPGA Monitor Program».

					ДКЗ1мп. 466539.001 ПЗ	Лист
						3
Зм.	Лист	№ докум.	Підпис	Дата		

Саме через термінал програмного «Intel FPGA Monitor Program» і буде здійснюватися контроль семисегментних індикаторів.

Такий вибір теми дозволить виконати освітницьку ціль даного курсового проєкту, адже буде реалізована ціла система на кристалі. Вона скрадатиметься з самостійно розробленого модуля, мікропроцесора та об'єднуючих компонентів.

					ДКЗ1мп. 466539.001 ПЗ	Лист
						4
Зм.	Лист	№ докум.	Підпис	Дата		

РОЗДІЛ 1. ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Керування семисегментним індикатором.

В даному проекті я використовую семисегментного індикатор, схема якого зображена на Рис. 3.

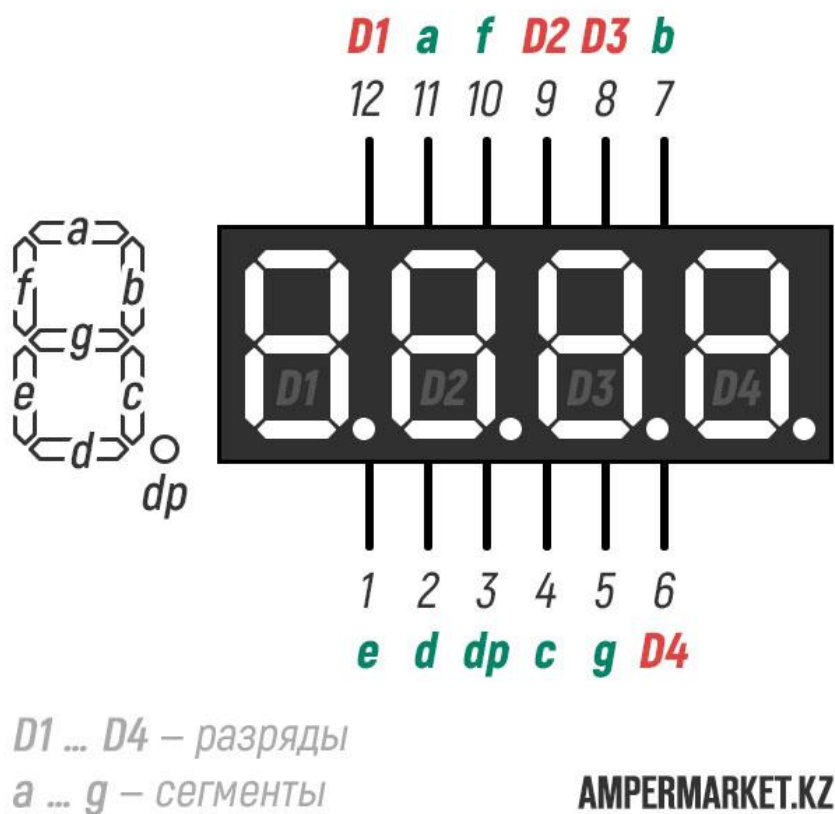


Рис. 3. Схема семисегментного індикатора, що буде використовуватись.

Зауважу, що це індикатор з спільним катодом. Це означає, що для загорання світлодіодів, слід подавати на «логічну 1» на піни a, b, c, d, e, f, g та при цьому гарантувати «логічний 0» на об'єднаних катодах D1, D2, D3, D4.

Для того, щоб відображати різні значення на кожному з індикаторів, слід швидко перемикає між ними. В моєму проекті задачу контролю семисегментним індикатором на себе бере оригінальний модуль, формований на базі компонентів плати програмованої логіки DE10-nano. Цей модуль буде

отримувати 16 біт інформації, які буде відображати на індикаторах в шіснадцятковій системі числення.

1.2 Робота з терміналом середовища “Intel FPGA Monitor Program”

«Intel FPGA Monitor Program» – це програмне середовище, що дозволяє працювати з мікропроцесорним ядром NiosII, яке в свою чергу інтегроване в мікросхему системи на кристалі на платі DE10-Nano. Цією мікросхемою є чіп 5CSEBA6U23I7.

Для роботи використовується стандартна бібліотека мови програмування «C» - бібліотека «stdio.h».

Зазначу, що будувати програмне забезпечення слід з врахуванням адреси, яка належатиме модулю керування семисигментним індикатором. Для компілювання можна використовувати те ж середовище «Intel FPGA Monitor Program», от тільки для цього слід встановити систему «Windows Subsystem for Linux», а також додати деякі «Шляхи» в «Змінні оточення системи Windows».

Тільки після цього, появляється можливість компілювати файл з розширенням «.c». Отриманий бінарний файл з інструкціями для ядра і буде завантажений в модуль пам’яті і стане «пам’яттю програм» для мікропроцесорного ядра. Компіляція зображена на Рис. 4.



Рис. 4. Компіляція фалу «main.c».

РОЗДІЛ 2. СТРУКТУРНА СХЕМА ПРИСТРОЮ

2.1 Структурна схема пристрою та її опис

На Рис. 5 зображена схематичне представлення пристрою.

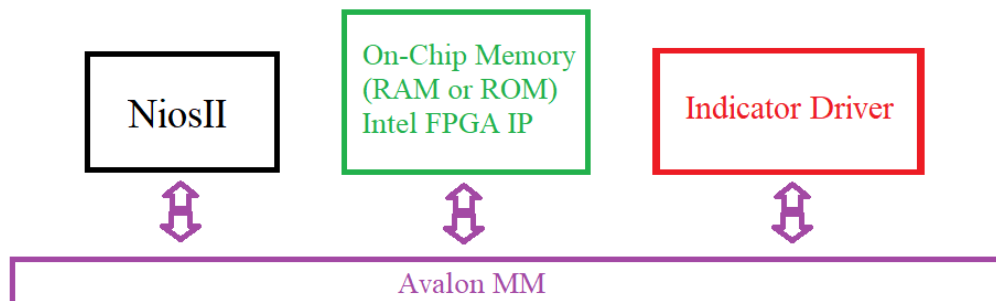


Рис. 5. Схематичне зображення системи на кристалі.

Компонент «NiosII» – це вбудоване в систему на кристалі мікропроцесорне ядро.

Компонент «On-Chip Memory (RAM or ROM) Intel FPGA IP» – це блок, що утворюється з компонентів пам'яті всередині системи на кристалі. Він утворюється об'єднанням невеликих комірок в одну більшу систему пам'яті.

Компонент «Indicator Driver» – це оригінальний блок, що HDL описується мовою «verilog». Цей модуль синтезується з різноманітних складових програмованої логіки системи на кристалі. Лістинг коду цього модуля зображено на Рис. 7:

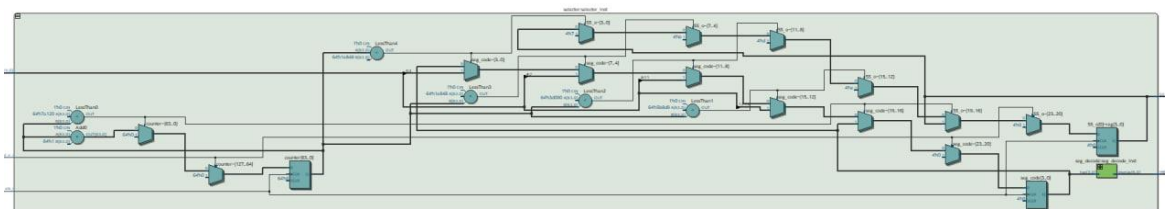


Рис. 6. Вигляд модуля «Indicator Driver» в утиліті «RLT Viewer»

```

1  module selector (
2      input clk_i,
3      input rst_n_i,
4      input [15:0] reg_16_i,
5      output wire [6:0] seg_display_o,
6      output reg [3:0] SS_o
7  );
8
9      parameter period = 50 * 10 * 1000;
10
11     reg [63:0] counter;
12     reg [3:0] seg_code;
13
14     always @(posedge clk_i) begin
15         if(rst_n_i == 0) begin
16             counter <= 64'b0;
17             SS_o = 4'b0;
18             seg_code <= 4'b0;
19         end else begin
20             counter <= counter + 1;
21             if (counter > period) begin
22                 counter <= 64'b0;
23             end else if(counter > period / 4 * 3) begin
24                 SS_o <= 4'b0111;
25                 seg_code <= reg_16_i[15:12];
26             end else if(counter > period / 4 * 2) begin
27                 SS_o <= 4'b1011;
28                 seg_code <= reg_16_i[11:8];
29             end else if(counter > period / 4) begin
30                 seg_code <= reg_16_i[7:4];
31                 SS_o <= 4'b1101;
32             end else if(counter < period / 4) begin
33                 SS_o <= 4'b1110;
34                 seg_code <= reg_16_i[3:0];
35             end
36         end
37     end
38
39
40     seg_decode seg_decode_inst(seg_code, seg_display_o);
41
42 endmodule

```

Рис. 7. Лістинг коду на мові «verilog», що описує модуль «Indicator Driver».

РОЗДІЛ 3. СТВОРЕННЯ ПРИСТРОЮ

3.1. Створення та симуляція модуля «Indicator Driver»

Під час написання файлу «selector.v», для того, щоб перевіряти модуль, було написано допоміжний файл «tb_for_selector.v» (рис. 8) та файл «sim.do» (рис. 9) для запуску симуляції в середовищі «ModelSim».

```
tb_for_selector.v
1 // мс - наносекунди ns - мілісекунди SI - семисегментний індикатор
2 // В цьому файлі проведемо симуляцію свого модуля, що керує СИ. Цей модуль тактується частотою 50 МГц і керує чотирма СИ.
3 // За один період він по черговому драйвить всі 4 СИ. Період його роботи це 50 000 тактів, на частоті в 50 МГц це 10 мс. Таким чином кожен СИ драйвиться 2,5 мс, а частота драйву ...
4 // складає 100 Гц. Для сприйняття людиною цього достатньо.
5 // Буду симулювати протягом однієї секунди. На цьому проміжку буде два ресети.
6 // Встановлюю частоту координати
7 `timescale 1ns/100ps
8 module tb_for_selector ();
9     reg clk_i = 0;
10    reg rst_n_i;
11    reg [15:0] reg_16_i;
12    wire [5:0] seg_display_o;
13    wire [3:0] SS_o;
14    parameter P = 20; // Період становить 20 мс, тобто частота 50 МГц
15    parameter Tms = P * 50 * 1000; // Часовий проміжок в одну мс
16    // Екземпляр модуля декодера
17    selector_inst(
18        .clk_i(clk_i),
19        .rst_n_i(rst_n_i),
20        .reg_16_i(reg_16_i),
21        .seg_display_o(seg_display_o),
22        .SS_o(SS_o)
23    );
24    initial # (Tms * 100) $stop; // Моделювання триватиме 100мс
25    initial forever # (P / 2) clk_i = ~clk_i; // Формуємо тактовий сигнал
26    // Гереування ресетів
27    initial begin
28        // Перший ресет на 10мс від початку симуляції.
29        # (Tms * 10)
30        # (Tms) rst_n_i = 1;
31        # (Tms) rst_n_i = 0;
32        # (Tms) rst_n_i = 1;
33
34        // Тут вже пройшло 13 мс. Другий ресет на 60 мс
35        # (Tms * 47)
36        # (Tms) rst_n_i = 1;
37        # (Tms) rst_n_i = 0;
38        # (Tms) rst_n_i = 1;
39    end
40    // Гереування інформації для СИ
41    initial begin
42        // на 20 мс
43        # (Tms * 20)
44        reg_16_i = 16'b0;
45        // на 40мс
46        # (Tms * 20)
47        reg_16_i = 16'h1234;
48        // на 60мс
49        # (Tms * 20)
50        reg_16_i = 16'h4321;
51        // на 80мс
52        # (Tms * 20)
53        reg_16_i = 16'hFEDC;
54    end
55 endmodule
```

Рис.8. Лістинг файлу «tb_for_selector.v»

					ДКЗ1мп. 466539.001 ПЗ	Лист
						9
Зм.	Лист	№ докум.	Підпис	Дата		


```

sim.do
1  if { [file exists "work"] } { vdel -all }
2
3  vlib work
4
5  vlog C:/Users/bimee/OneDrive/SoC/Course_Task/Verilog_files/selector.v
6  vlog C:/Users/bimee/OneDrive/SoC/Course_Task/Verilog_files/seg_decode.v
7  vlog C:/Users/bimee/OneDrive/SoC/Course_Task/Verilog_Testbench/tb_for_selector.v
8
9  vsim work.tb_for_selector
10
11
12  add wave -radix binary sim:/tb_for_selector/clk_i
13  add wave -radix binary sim:/tb_for_selector/rst_n_i
14
15  add wave -radix binary sim:/tb_for_selector/seg_display_o
16  add wave -radix binary sim:/tb_for_selector/SS_o
17
18  add wave -radix hexadecimal sim:/tb_for_selector/reg_16_i
19
20
21  onbreak resume
22  run -all
23  scale 1ns
24  wave zoom full

```

Рис.9. Лістинг файлу «sim.do»

Запустивши симуляцію командою «vsim -do sim.do» можна побачити часову діаграму зображену на рис. 10.

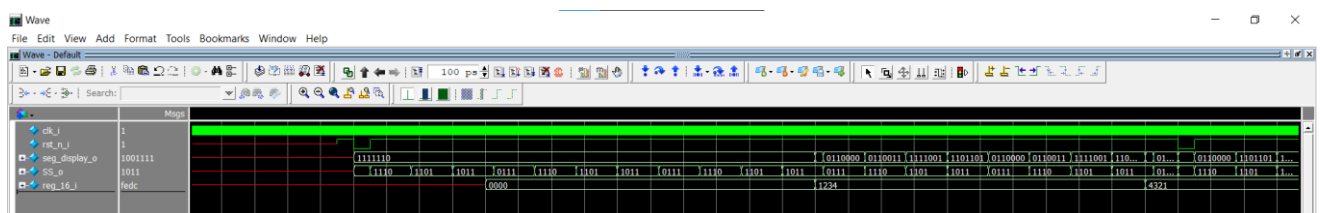


Рис. 10. Симуляція роботи модуля «selector», який і є компонентом «Indicator Driver».

Зауважу, що кожен біт «Slave select» є катодом і тому світиться саме той індикатор чий «Slave select» рівний нулю, тому що саме тоді через світлодіоди протікає струм. Щоб зрозуміти якому біту відповідає який світлодіод, дивіться рис. 11.

					ДКЗ1мп. 466539.001 ПЗ	Лист
						10
Зм.	Лист	№ докум.	Підпис	Дата		

```

1  module seg_decode (hex, display);
2
3      input [3:0] hex;
4      output reg [6:0] display;
5
6      /* - 0 -
7       * 5 | | 1
8       * - 6 -
9       * 4 | | 2
10      * - 3 -
11      */
12
13      always @ (hex)
14      case (hex)
15          4'h0: display = 7'b1111110;
16          4'h1: display = 7'b0110000;
17          4'h2: display = 7'b1101101;
18          4'h3: display = 7'b1111001;
19          4'h4: display = 7'b0110011;
20          4'h5: display = 7'b1011011;
21          4'h6: display = 7'b1011111;
22          4'h7: display = 7'b1110000;
23          4'h8: display = 7'b1111111;
24          4'h9: display = 7'b1110011;
25          4'hA: display = 7'b1110111;
26          4'hb: display = 7'b0011111;
27          4'hC: display = 7'b1001110;
28          4'hd: display = 7'b0111101;
29          4'hE: display = 7'b1001111;
30          4'hF: display = 7'b1000111;
31      endcase
32
33  endmodule

```

Рис.11. Лістинг файлу «seg_decode.v»

3.2. Проєкт в середовищі «Quartus»

При створенні проєкту в середовищі «Quartus», першим ділом було додано файли як на Рис. 12.

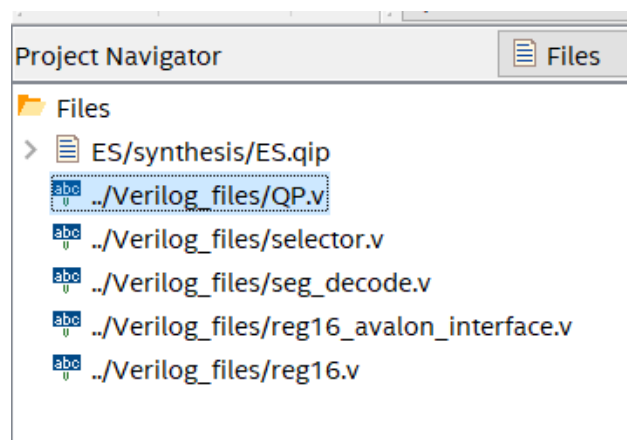


Рис.12. Файли проєкту в середовищі «Quartus».

Файлом верхнього рівня ієрархії є файл «QP.v» (Рис. 13), тобто «Quartus Project». Він містить мій оригінальний модуль «selector» та модуль, що описується в файлі «ES.qip» (Рис. 14).

```

1  module QP (
2      input CLOCK_50_i,
3      input KEY_rst_n_i,
4      output [6:0] SEG_o,
5      output [3:0] SS_o
6  );
7
8      wire [31:0] reg32;
9
10     ES ES_inst (
11         .clk_clk(CLOCK_50_i),
12         .reset_reset_n(KEY_rst_n_i),
13         .to_leds_readdata(reg32)
14     );
15
16     selector selector_inst (
17         .clk_i(CLOCK_50_i),
18         .rst_n_i(KEY_rst_n_i),
19         .reg_16_i(reg32[15:0]),
20         .seg_display_o(SEG_o),
21         .SS_o(SS_o)
22     );
23
24 endmodule

```

Рис. 13. Файл «QP.v».

System Contents Address Map Interconnect Requirements									
System: ES Path: clk_0									
Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
<input checked="" type="checkbox"/>		clk_0	Clock Source		exported				
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk					
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	reset					
<input checked="" type="checkbox"/>		clk	Clock Output	Double-click to export	clk_0				
<input checked="" type="checkbox"/>		clk_reset	Reset Output	Double-click to export					
<input checked="" type="checkbox"/>		nios2_qsys_0	Nios II (Classic) Processor						
<input checked="" type="checkbox"/>		clk	Clock Input	Double-click to export	clk_0				
<input checked="" type="checkbox"/>		reset_n	Reset Input	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		d_irq	Interrupt Receiver	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		jtag_debug_module_reset	Reset Output	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		jtag_debug_module	Avalon Memory Mapped Slave	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		custom_instruction_master	Custom Instruction Master	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM)...						
<input checked="" type="checkbox"/>		clk1	Clock Input	Double-click to export	clk_0				
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]				
<input checked="" type="checkbox"/>		reset1	Reset Input	Double-click to export	[clk1]				
<input checked="" type="checkbox"/>		reg32_avalon_interface_0	reg32_avalon_interface_compo...						
<input checked="" type="checkbox"/>		clock_reset	Reset Input	Double-click to export	[clock_sink]				
<input checked="" type="checkbox"/>		avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]				
<input checked="" type="checkbox"/>		clock_sink	Clock Input	Double-click to export	clk_0				
<input checked="" type="checkbox"/>		conduit_end	Conduit	to_leds	[clock_sink]				
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART Intel FPGA IP						
<input checked="" type="checkbox"/>		clk	Clock Input	Double-click to export	clk_0				
<input checked="" type="checkbox"/>		reset	Reset Input	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]				
<input checked="" type="checkbox"/>		irq	Interrupt Sender	Double-click to export	[clk]				

Рис. 14. Файл «ES.qip».

Як видно з рис. 14, модуль ES (Embedded System) є файлом конфігурації системи на кристалів для мікросхеми FPGA. Саме в цьому модулі компоненти і були налаштовані, з'єднані шиною «Avalon MM» та були встановлені їх адреси.

Файл «reg32.v» (рис. 15) – виступає в ролі 4 байт пам'яті яка спільна для ядра та оригінального модуля. Було би достатньо і 16 біт, але оскільки шина даних «Avalon MM» має 32 біти, то робити модуль меншим за 32 немає сенсу.

```

1  module reg32 (clock, resetn, D, byteenable, Q);
2      input clock, resetn;
3      input [3:0] byteenable;
4      input [31:0] D;
5      output reg [31:0] Q;
6
7      always @(posedge clock) begin
8          if (!resetn) Q <= 32'b0;
9          else begin
10             if (byteenable[0]) Q[7:0] <= D[7:0];
11             if (byteenable[1]) Q[15:8] <= D[15:8];
12             if (byteenable[2]) Q[23:16] <= D[23:16];
13             if (byteenable[3]) Q[31:24] <= D[31:24];
14         end
15     end
16 endmodule

```

Рис. 15. Файл «reg32.v»

Файл «reg32_avalon_interface.v» (рис. 16) – є файлом, що забезпечує необхідні виходи для підключення шини «Avalon MM» до модуля «reg32». Електричні схеми обох модулів зображені на рис. 17.

```

reg32_avalon_interface.v
1 module reg32_avalon_interface (
2     clock,
3     resetn,
4     writedata,
5     readdata,
6     write,
7     read,
8     byteenable,
9     chipselect,
10    Q_export
11 );
12
13    // signals for connecting to the Avalon fabric
14    input clock, resetn, read, write, chipselect;
15    input [3:0] byteenable;
16    input [31:0] writedata;
17    output [31:0] readdata;
18
19    // signal for exporting register contents
20    // outside of the embedded system
21    output [31:0] Q_export;
22    wire [3:0] local_byteenable;
23    wire [31:0] to_reg, from_reg;
24
25    assign to_reg = writedata;
26    assign local_byteenable = (chipselect & write) ? byteenable : 4'd0;
27
28    reg32 U1 (
29        .clock(clock),
30        .resetn(resetn),
31        .D(to_reg),
32        .byteenable(local_byteenable),
33        .Q(from_reg)
34    );
35
36    assign readdata = from_reg;
37    assign Q_export = from_reg;
38
39    endmodule

```

Рис. 16. Файл «reg32_avalon_interface.v»

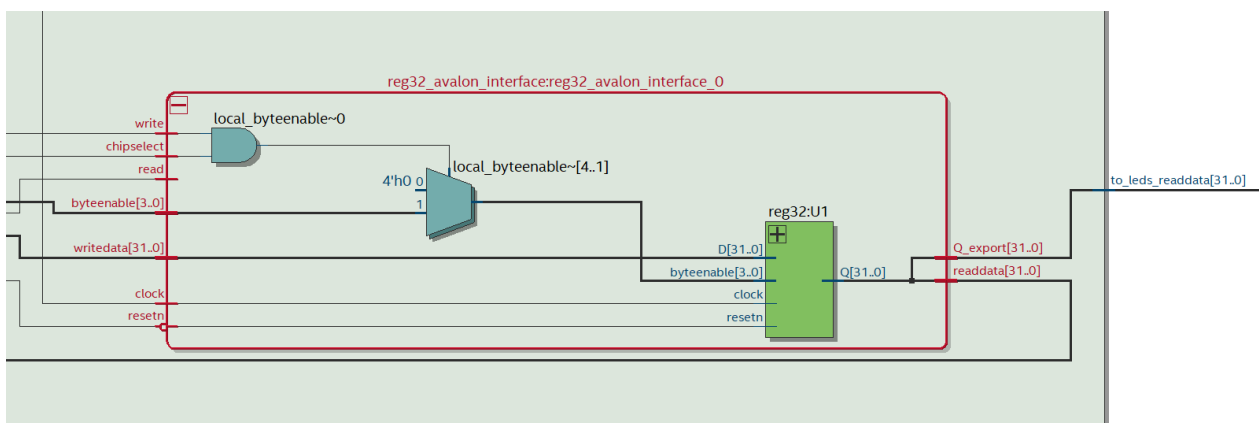
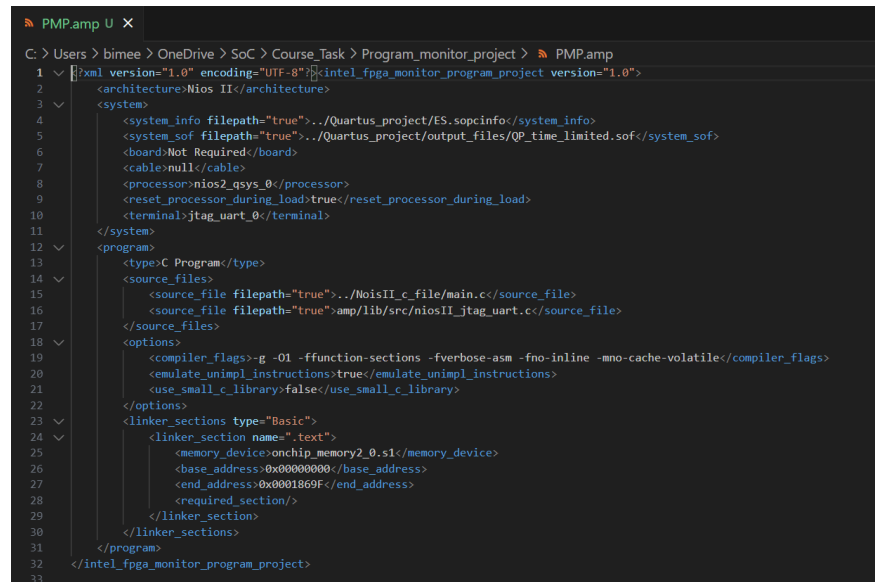


Рис. 17. Електричні схеми модулів «reg32_avalon_interface.v» та «reg32.v»

3.3. Проєкт в середовищі «Intel FPGA Monitor Program»

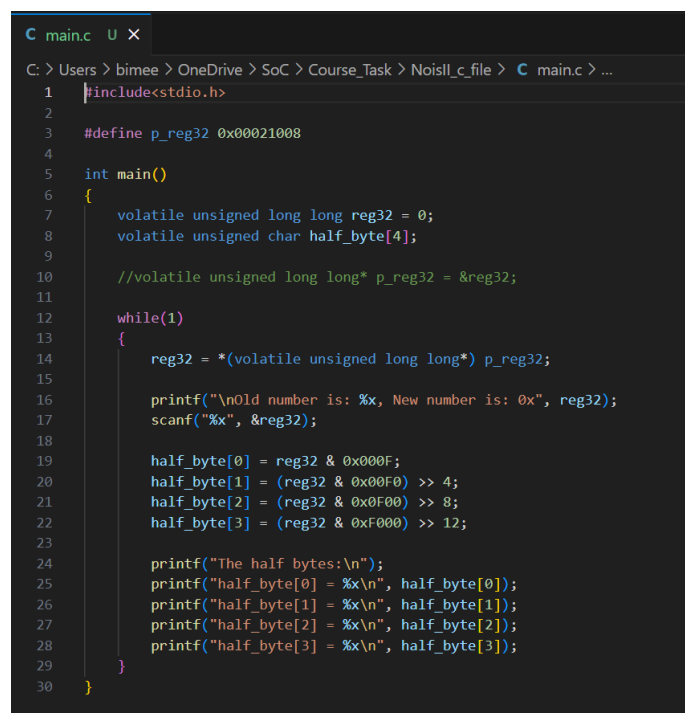
Для того, щоб завантажити код програми для процесора NoisII слід створити проєкт в середовищі «Intel FPGA Monitor Program». Ось якими параметрами він володіє (рис. 18):



```
PMP.ampr X
C:\Users> bimee > OneDrive > SoC > Course_Task > Program_monitor_project > PMP.ampr
1 <?xml version="1.0" encoding="UTF-8"?><intel_fpga_monitor_program_project version="1.0">
2 <architecture>Nios II</architecture>
3 <system>
4 <system_info filepath="true">../Quartus_project/ES.sopcinfo</system_info>
5 <system_sof filepath="true">../Quartus_project/output_files/QP_time_limited.sof</system_sof>
6 <board>Not_Required</board>
7 <cable>null</cable>
8 <processor>nios2_qsys_0</processor>
9 <reset_processor_during_load>true</reset_processor_during_load>
10 <terminal>jtag_uart_0</terminal>
11 </system>
12 <program>
13 <type>C_Program</type>
14 <source_files>
15 <source_file filepath="true">../NoisII_c_file/main.c</source_file>
16 <source_file filepath="true">../amp/lib/src/niosII_jtag_uart.c</source_file>
17 </source_files>
18 <options>
19 <compiler_flags>-g -O1 -ffunction-sections -fverbose-asm -fno-inline -mno-cache-volatile</compiler_flags>
20 <emulate_unimpl_instructions>true</emulate_unimpl_instructions>
21 <use_small_c_library>false</use_small_c_library>
22 </options>
23 <linker_sections type="Basic">
24 <linker_section name=".text">
25 <memory_device>onchip_memory2_0.s1</memory_device>
26 <base_address>0x00000000</base_address>
27 <end_address>0x0001869F</end_address>
28 <required_section>
29 </linker_section>
30 </linker_sections>
31 </program>
32 </intel_fpga_monitor_program_project>
33
```

Рис. 18. Файл «PMP.ampr», що і визначає налаштування проєкту.

А ось і алгоритм роботи процесора (рис. 19):



```
C main.c U X
C:\Users> bimee > OneDrive > SoC > Course_Task > NoisII_c_file > C main.c > ...
1 #include<stdio.h>
2
3 #define p_reg32 0x00021008
4
5 int main()
6 {
7     volatile unsigned long long reg32 = 0;
8     volatile unsigned char half_byte[4];
9
10    //volatile unsigned long long* p_reg32 = &reg32;
11
12    while(1)
13    {
14        reg32 = *(volatile unsigned long long*) p_reg32;
15
16        printf("\nOld number is: %x, New number is: 0x", reg32);
17        scanf("%x", &reg32);
18
19        half_byte[0] = reg32 & 0x000F;
20        half_byte[1] = (reg32 & 0x00F0) >> 4;
21        half_byte[2] = (reg32 & 0x0F00) >> 8;
22        half_byte[3] = (reg32 & 0xF000) >> 12;
23
24        printf("The half bytes:\n");
25        printf("half_byte[0] = %x\n", half_byte[0]);
26        printf("half_byte[1] = %x\n", half_byte[1]);
27        printf("half_byte[2] = %x\n", half_byte[2]);
28        printf("half_byte[3] = %x\n", half_byte[3]);
29    }
30 }
```

Рис. 19. Файл «main.c», що задає алгоритм роботи процесору.

Оскільки в мене під час виконання курсового проєкту виникли проблеми з платою DE10-nano (рис. 20), то я дещо зміню «main.c» та продемонструю його роботу у відриві від решти компонентів системи (рис. 21).

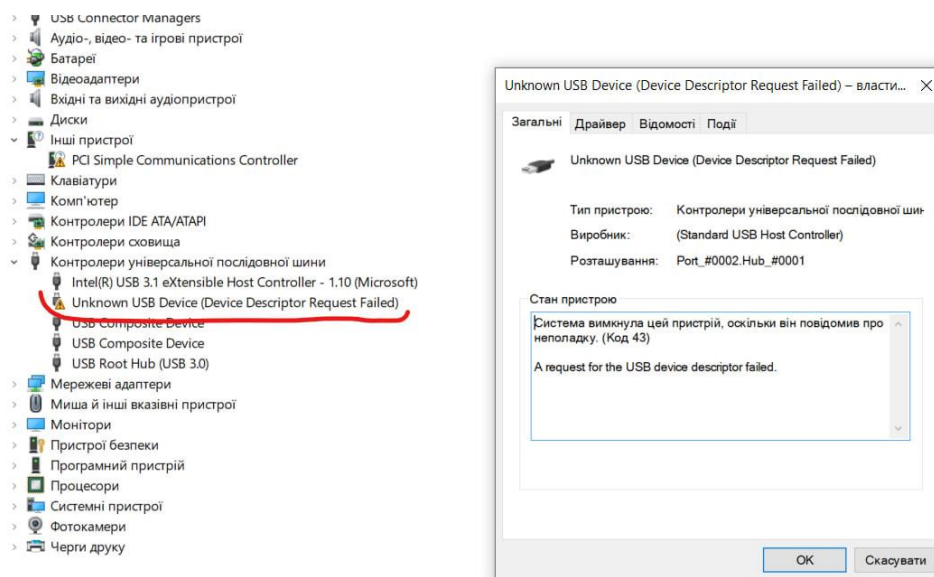


Рис. 20. Вихід з ладу плати DE10-nano.

```

Windows PowerShell
PS C:\Users\bimee\OneDrive\SoC\Course_Task\NoisII_c_file> .\run.bat

C:\Users\bimee\OneDrive\SoC\Course_Task\NoisII_c_file>gcc main.c -o main.exe

C:\Users\bimee\OneDrive\SoC\Course_Task\NoisII_c_file>.\main.exe

old number is: 0, New number is: 0x1234
The half bytes:
half_byte[0] = 4
half_byte[1] = 3
half_byte[2] = 2
half_byte[3] = 1

old number is: 1234, New number is: 0xF12A
The half bytes:
half_byte[0] = a
half_byte[1] = 2
half_byte[2] = 1
half_byte[3] = f

old number is: f12a, New number is: 0x
    
```

Рис. 21. Демонстрація роботи «main.c» у відриві від решти компонентів системи на кристалі.

ВИСНОВОК

В межах даного курсового проєкту було розроблено систему на кристалі, котра базується на платі DE10-nano. Ця система використовує взаємодію процесорне ядро та засоби відладки для того, щоб обробляти інформацію, що надходить з терміналу утиліти «Intel FPGA Monitor Program». Також є оригінальний компонент, що отримує дані від ядра через шину «Avalon MM» та у відповідності до отриманої інформації керує сукупністю семисигментних індикаторів. Система містить й інші компоненти як от пам'ять чи порти PІО.

Нажаль, мені не вдалось перевірити всі елементи в взаємодії та додати такий експеримент в пояснювальну записку, але враховуючи досвід запуску подібних систем в межах іншої дисципліни, вважаю, що пристрій розроблено без грубих помилок. Також присутня перевірка оригінального модуля за допомогою засобу симуляції «ModelSim» та перевірка програмного забезпечення процесора NiosII у відриві від решти компонентів.

Враховуючи такий результат, вважаю, що курсовий проєкт заслуговує оцінки «достатньо»!

					ДКЗ1мп. 466539.001 ПЗ	Лист
						17
Зм.	Лист	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DE10-Nano User Manual [електроний ресурс] – Режим доступу:
https://ftp.intel.com/Public/Pub/fpgaup/pub/Intel_Material/Boards/DE10-Nano/DE10_Nano_User_Manual.pdf
2. DE10-Nano Board [електроний ресурс] – Режим доступу:
https://www.mikrocontroller.net/attachment/327098/de10-nano_a0.pdf
3. Introduction to the Intel Nios II Soft Processor [електроний ресурс] – Режим доступу:
<https://classroom.google.com/c/ODU2NzY2MzUyMjFa/m/MTY5MTMxNTg0NTcx/details>

					ДКЗ1мп. 466539.001 ПЗ	Лист
						18
Зм.	Лист	№ докум.	Підпис	Дата		